BAB 2
TINJAUAN PUSTAKA

2.1 Profil Institusi

Profil Institusi adalah untuk mengetahui keadaan institusi diantaranya adalah sejarah berdirinya, struktur organisasi serta visi dan misi.

2.1.1 Sejarah Cafe La Ponyo

Cafe La Ponyo berdiri pada awal tahun 2009, Cafe La Ponyo merupakan bagian dari Rumah makan Ponyo dan usaha yang bergerak dalam bidang makanan ini beralamat di Jalan Raya Cinunuk No. 186. Cafe La Ponyo menghadirkan konsep modern gaya elegan minimalis, menyediakan berbagai macam menu *sundanese*, *western*, dan *seafood*, Cafe La Ponyo memberikan fasilitas *Hot Spot* guna memberikan rasa kepuasan dan kenyamanan bagi pelanggan setianya. Fasilitas internet gratis tanpa batas dengan kecepatan tinggi telah menjadikan Cafe La Ponyo sebagai salah satu tempat paling favorit bagi anak muda dan kaum profesional lainnya. Keanekaragaman menu dan jenis makanan juga terdiri dari keseimbangan dan kesadaran akan asupan nutrisi, yang tentunya dibalut dengan sentuhan citarasa tinggi dan segudang pengalaman memasak para koki kafe ini.

2.1.2 Visi dan Misi

Visi dari Cafe La Ponyo adalah untuk mengembangkan usaha atau bisnis dalam bidang makanan dan menjadi cafe pilihan pertama dan utama di Bandung.

Adapun misi dari Cafe La Ponyo adalah sebagai berikut :

1. Menjadikan Cafe La Ponyo sebagai tempat yang nyaman bagi pelanggan.
2. Menyajikan makanan dan minuman yang berpotensi menjadi wisata kuliner Nasional.
3. Memberi pelayanan yang dapat memuaskan pelanggan baik dari segi keramahan, maupun kebersihan tempat dan makanan.

2.1.3 Logo Cafe La Ponyo

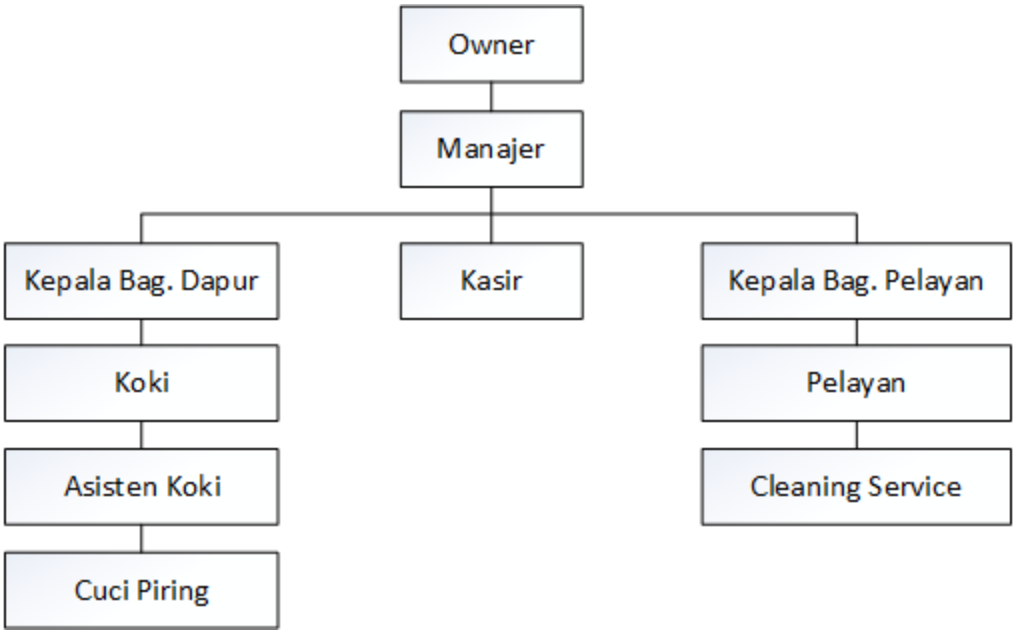
Berikut merupakan logo dari Cafe La Ponyo seperti yang terdapat pada gambar 2.1.



Gambar 2.1 Logo Cafe La Ponyo

2.1.4 Struktur Organisasi

Berikut merupakan Struktur Organisasi dari Cafe La Ponyo seperti yang terdapat pada gambar 2.2.



Gambar 2.2 Struktur Organisasi Cafe La Ponyo

2.2 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam pembangunan aplikasi pemesanan menu makanan.

2.2.1 Sistem Informasi

Dalam pengertian sistem informasi ada baiknya terlebih dahulu mengerti dari sistem dan informasi. Dengan memahami defenisi sistem dan informasi akan dapat diperoleh penjelasan yang baik tentang sistem informasi.

2.2.1.1 Pengertian Sistem

Istilah sistem merupakan suatu istilah yang tidak asing lagi dan banyak digunakan secara luas pada lembaga-lembaga atau bidang-bidang ilmu pengetahuan dan teknologi. Istilah sistem berasal dari bahasa Yunani yaitu *Sistema*. Ditinjau dari asal katanya, sistem berarati sekumpulan objek yang bekerja bersama-sama untuk menghasilkan suatu kesatuan metode yang digabungkan dan diatur sedemikian rupa yang berfungsi mencapai tujuan. Suatu sistem dapat terdiri dari beberapa subsistem yang saling berhubungan membentuk suatu kesatuan sehingga sasaran atau tujuan sistem tersebut dapat tercapai. Pada umumnya ciri-ciri sistem bertujuan, punya batas, terbuka, tersusun dari subsistem, dan saling berkaitan dan saling ketergantungan, merupakan suatu kebulatan yang utuh, melakukan kegiatan transformasi, ada mekanisme kontrol, dan memiliki kemampuan mengatur dan menyesuaikan diri sendiri[1].

2.2.1.2 Pengertian Informasi

Informasi merupakan data yang telah diproses, transformasi data, sehingga berubah bentuk menjadi informasi. Informasi adalah data yang telah diambil kembali, diolah, atau sebaliknya digunakan untuk tujuan *informative*, argumentasi, ataupun sebagai dasar untuk pengambilan keputusan. Informasi ini adalah hasil proses data yang bentuknya kurang berguna menjadi data yang berguna. Menurut Jogiyanto Hartono (1999, hal :696)[1] kualitas suatu informasi tergantung dari tiga hal, yaitu informasi harus akurat, tepat pada waktunya dan relevan.

1. Akurat berarti informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan.
2. Tepat pada waktunya berarti informasi yang datang pada penerima tidak boleh terlambat.
3. Relevan berarti informasi tersebut mempunyai manfaat untuk pemakainya.

Kegunaan informasi itu sendiri adalah untuk memberitahukan ke pengguna informasi mengenai suatu masalah agar pengguna informasi lebih dapat menguasai masalah yang dihadapinya. Informasi juga ketidakpastian tentang suatu masalah yang dapat digunakan untuk memilih resiko yang paling kecil dan keuntungan yang besar dalam pemilihan alternatif bagi suatu proses pengambilan keputusan[1].

2.2.1.3 Pengertian Sistem Informasi

Dari definisi sistem dan informasi dapat disimpulkan bahwa sistem informasi adalah suatu sistem di dalam suatu sistem organisasi yang merupakan kombinasi dari orang-orang, fasilitas, teknologi, media, prosedur-prosedur, dan pengendalian, yang ditujukan untuk mendapatkan jalur komunikasi penting, memproses tipe transaksi rutin tertentu, memberi sinyal kepada manajemen dan yang lainnya terhadap kejadian-kejadian internal dan eksternal yang penting dan menyediakan sutau dasar informasi untuk pengambilan keputusan yang cerdik (Jogiyanto Hartono, 1999)[1].

2.2.1.4 Peranan Komputer dalam Sistem Informasi

Komputer mempunyai peranan penting untuk membantu manusia dalam menyelesaikan dan memecahkan masalah yang dihadapinya. Setiap pemecahan persoalan yang dilakukan mansia, hendaknya mempunyai dua unsur, yaitu :

1. Adanya fakta dan data.
2. Bagaimana proses pemecahannya.

Jika pengelohan data dilakukan secara manual tidak akan menjamin ketelitian serta kebenaran hasil atau informasi yang diinginkan. Masalah pendataan dan pengelohan data dapat dipecahkan dengan mudah dalam waktu singkat dengan menggunakan alat bantu komputer.

Dalam hal inilah perlu diperhatikan suatu sistem pengolahan data yang cepat dan tepat untuk memberikan informasi yang sangat menjamin benarnya keputusan yang telah ditetapkan. Dilihat dari aspek praktisnya, maka tidak jarang seseorang terlambat atau belum selesai didalam mengambil keputusan dan mendapat informasi yang cepat dalam pengolahan komputer. Peranan komputer dalam suatu sistem informasi sangatlah penting. Hal ini diakibatkan besar dan banyaknya data

yang akan diolah dengan beraneka ragam jenis data. Disinilah komputer memegang peranan penting untuk melakukan pengolahan data yang banyak dan beraneka ragam informasi dapat diperoleh dengan cepat dan tepat.

2.2.2 Basis Data

Basis data (bahasa Inggris: *database*), atau sering pula dieja basis data, adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) basis data disebut sistem manajemen basis data (*database management system*, DBMS). Sistem basis data dipelajari dalam ilmu informasi.Istilah "basis data" berawal dari ilmu komputer. Meskipun kemudian artinya semakin luas, memasukkan hal-hal di luar bidang elektronika, artikel ini mengenai basis data komputer. Catatan yang mirip dengan basis data sebenarnya sudah ada sebelum revolusi industri yaitu dalam bentuk buku besar, kuitansi dan kumpulan data yang berhubungan dengan bisnis[2].

Istilah basis data mengacu pada koleksi dari data-data yang saling berhubungan, dan perangkat lunaknya seharusnya mengacu sebagai sistem manajemen basis data (*database management system*/DBMS). Jika konteksnya sudah jelas, banyak administrator dan *programmer* menggunakan istilah basis data untuk kedua arti tersebut[2].

Konsep dasar dari basis data adalah kumpulan dari catatan-catatan, atau potongan dari pengetahuan. Sebuah basis data memiliki penjelasan terstruktur dari jenis fakta yang tersimpan di dalamnya: penjelasan ini disebut skema. Skema menggambarkan obyek yang diwakili suatu basis data, dan hubungan di antara obyek tersebut. Ada banyak cara untuk mengorganisasi skema, atau memodelkan struktur basis data: ini dikenal sebagai model basis data atau model data. Model yang umum digunakan sekarang adalah model relasional, yang menurut istilah layman mewakili semua informasi dalam bentuk tabel-tabel yang saling berhubungan dimana setiap tabel terdiri dari baris dan kolom (definisi yang sebenarnya menggunakan terminologi matematika). Dalam model ini, hubungan antar tabel diwakili denga menggunakan nilai yang sama antar tabel. Model yang

lain seperti model hierarkis dan model jaringan menggunakan cara yang lebih eksplisit untuk mewakili hubungan antar table[2].

2.2.2.1 Perancangan Basis Data

Perancangan basis data dibutuhkan agar diperoleh suatu sistem yang sesuai dengan apa yang diinginkan, dimana dapat melalui tahapan berikut :

a. *Entity Relationship Diagram*

ERD atau biasa disebut dengan diagram E-R secara grafis menggambarkan isi sebuah database. Diagram ini memiliki dua komponen utama yaitu entity dan relasi. Untuk melambangkan fungsi diatas maka digunakan simbol-simbol yang bisa dilihat pada daftar simbol[10].

b. Normalisasi

Normalisasi merupakan tahapan perancangan dalam membangun basis data relasional yangtidak secara langsung berkaitan dengan model data tetapi menerapkan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal. Adapun langkah-langkah untuk menghasilkan struktur tabel yang normal diantaranya[10] yaitu :

1. Bentuk *Unnormallized*

   Bentuk *unnormal* adalah bentuk tabel dengan mencantumkan semua field data yang ada.

2. Bentuk Normal Pertama / 1 NF (*First Normal Form*)

   Bentuk normal pertama terpenuhi jika sebuah tabel tidak memiliki atribut bernilai banyak (*Multivalued Atribute*) atau lebih dari satu atribut dengan domain nilai yang sama.

3. Bentuk Normal Kedua / 2 NF (*Second Normal Form*)

   Bentuk normal kedua terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam *key primer* memiliki ketergantungan fungsional pada *key primer* secara utuh.

4. Bentuk Normal Ketiga / 3 NF (*Third Normal Form*)

   Bentuk normal ketiga terpenuhi jika dan hanya jika semua atribut bukan kunci memiliki dependensi transitif terhadap kunci primer.

5. *Boyce-Codd Normal Form* (BCNF)

   Suatu tabel dikatakan berada pada BCNF jika dan hanya jika semua ketergantungan fungsional dengan notasi X→Y, maka X harus merupakan superkey pada tabel tersebut.

6. Bentuk Normal Keempat / 4 NF (Fourth Normal Form)

   Suatu tabel dikatakan berada pada normal keempat jika tidak mengandung dua atribut atau lebih yang bernilai banyak.

7. Bentuk Normal Kelima / 5 NF (Fifth Normal Form)

   Bentuk normal kelima berkaitan dengan ketergantungan relasi antar *table* (*join dependency*).

8. Relasi Tabel

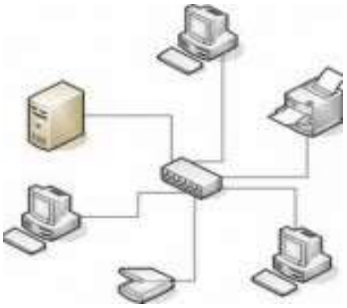
   Relasi tabel menunjukkan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Dari adanya relasi tabel tersebut terdapat kardinalitas relasi yang menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Terdapat empat kemungkinan kombinasi kardinalitas yaitu satu ke satu (one to one), satu ke banyak (one to many), banyak ke satu (many to one) dan banyak ke banyak (many to many) [10].

2.2.3 Sistem Jaringan Komputer

Dalam pengertian sistem jaringan komputer ada baiknya terlebih dahulu mengerti dari jaringan komputer. Dengan memahami defenisi jaringan komputer akan dapat diperoleh penjelasan yang baik tentang sistem jaringan komputer.

2.2.3.1 Pengertian Jaringan Komputer

Pengertian jaringan komputer secara sederhana adalah sekumpulan komputer berkomunikasi satu sama lain melalui media jaringan secara bersama-sama. Seperti yang terlihat pada gambar 2.3 berikut.



Gambar 2.3 Jaringan komputer

Techi Wire House mendefinisikan pengertian jaringan komputer yang telah dialih bahasakan bahwa Pengertian Jaringan komputer secara ekspilisit adalah kumpulan dua atau lebih komputer yang terhubung. Ketika komputer ini bergabung dalam jaringan, maka orang dapat berbagi *file* dan *peripheral* seperti *modem*, *printer*, *tape drive* cadangan, atau CD-ROM *drive*. Secara luas jaringan komputer bisa terhubung kebeberapa lokasi yang terhubung dengan menggunakan layanan yang tersedia dari Perusahaan penyedia Layanan Internet, sehingga jaringan komputer dapat mengirim *e-mail*, berbagi *link* ke *Internet global*, atau melakukan konferensi video secara *real time* dengan pengguna jarak jauh lainnya.[4]

Kesimpulannya pengertian jaringan Komputer adalah Sebuah jaringan komputer adalah sekelompok komputer atau lebih terhubung satu sama lain secara elektronik. Ini berarti bahwa Sebuah jaringan komputer, juga disebut hanya sebagai jaringan, terdiri dari dua atau lebih komputer, dan perangkat lainnya biasanya juga (seperti printer, hard drive eksternal, modem dan router), yang terhubung bersama sehingga mereka dapat berkomunikasi satu sama lain dan dengan demikian perintah pertukaran dan berbagi data, perangkat keras dan sumber daya lainnya. komputer dapat "berkomunikasi" satu sama lain artinya setiap komputer di jaringan dapat mengirimkan informasi ke yang lain[4].

2.2.3.2 Pengertian Sistem Jaringan Komputer

Sistem jaringan komputer adalah sekumpulan aturan dan peralatan sebagai satu kesatuan yang terdiri terdiri atas komputer-komputer yang didesain untuk dapat berbagi sumber daya (printer, CPU), berkomunikasi dan dapat mengakses informasi secara bersama[4].

Sebuah sistem jaringan komputer melibatkan dua buah komputer yang dihubungkan dengan menggunakan media online atau online telepon. Sedangkan sistem jaringan yang rumit atau kompleks tergantung pada imajinasi masing-masing. Khususnya, jaringan komputer berarti semua node seperti server (*server*), stasiun kerja (*workstation*), printer (printer) dan sebagainya dihubungkan satu sama lain dengan tujuan untuk berbagi informasi dan bahan. Dengan kata lain, informasi yang ada pada pengguna A dapat disebarkan kepada penguna lain dan begitu juga sebaliknya[4].

2.2.4 Object Oriented Programming (OOP)

OOP merupakan terobosan baru dalam pemrograman setelah pemrograman prosedural, pemrograman modular dan pemrograman abstraksi data. Ada beberapa konsep utama OOP antara lain :

Pemrograman berorientasi-objek menekankan konsep berikut:

1. ***Class*** : kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh 'class of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing.
2. **Objek** : membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
3. **Abstraksi** : kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.
4. **Enkapsulasi** : memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses

keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

5. ***Polimorfisme*** : melalui pengiriman pesan. tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai *polimorfisme* melalui penggunaan fungsi kelaspertama.
6. ***Inheritance*** - Mengatur polimorfisme dan enskapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada, objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa harus mengimplementasi ulang perilaku tersebut. Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut(terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manajer, sekretaris, petugas administrasi data dan lainnya. Misal manajer tersebut ingin memperoleh data dari bagian administrasi maka manajer tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager

tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek -objek yang ada karena setiap objek memiliki *job description*-nya sendiri[2].

2.2.5 UML

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari IBM.UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem[8].

2.2.5.1 Sejarah UML

Sampai era tahun 1990 puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dan sebagainya. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru

apabila kita bekerjasama dengan kelompok/perusahaan lain yang menggunakan metodologi yang berlainan.

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group[4].

2.2.5.2 Diagram UML

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. **Use Case Diagram** untuk memodelkan proses bisnis.
2. **Conceptual Diagram** untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. **Sequence Diagram** untuk memodelkan pengiriman pesan (message) antar objects
4. **Collaboration Diagram** untuk memodelkan interaksi antar objects.
5. **State Diagram** untuk memodelkan perilaku objects di dalam sistem.
6. **Activity Diagram** untuk memodelkan perilaku Use Cases dan objects di dalam system.
7. **Class Diagram** untuk memodelkan struktur kelas.
8. **Object Diagram** untuk memodelkan struktur object.
9. **Component Diagram** untuk memodelkan komponen object
10. **Deployment Diagram** untuk memodelkan distribusi aplikasi.

Berikut akan dijelaskan empat macam diagram yang paling sering digunakan dalam pembangunan aplikasi berorientasi object, yaitu use case diagram, sequence diagram, collaboration diagram, dan class diagram[4].

1. Use Case Diagram

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use case* diagram terdiri atas diagram untuk *use case* dan *actor*. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi. *Use case* merepresentasikan operasi-operasi yang dilakukan oleh *actor*. *Use case* digambarkan berbentuk *elips* dengan nama operasi dituliskan di dalamnnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *use case*[4].

2. *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan di sini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapatdilakukan sistem[4].

3. *Sequence Diagram*

Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi[4].

4. Collaboration Diagram

Collaboration diagram dipakai untuk memodelkan interaksi antar objek di dalam sistem. Berbeda dengan *sequence diagram* yang lebih menonjolkan kronologis dari operasi-operasi yang dilakukan, *collaboration diagram* lebih fokus pada pemahaman atas keseluruhan operasi yang dilakukan oleh objek[4].

5. *Class Diagram*

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan[4].

2.2.6 Software Pendukung

2.2.6.1 PHP *(Personal Home Page)*

PHP adalah bahasa pemorgraman yang memungkinkan para web developer untuk membuat aplikasi web yang dinamis dengan cepat. PHP merupakan singkatan dari "PHP: *Hypertext Preprocessor*". PHP ditulis dan diperkenalkan pertama kali sekitar tahun 1994 oleh Rasmus Lerdorf melalui situsnya untuk mengetahui siapa saja yang telah mengakses ringkasan *online*-nya[6].

PHP merupakan salah satu bahasa script yang terbilang baru dan tersedia secara bebas dan masih memungkinkan untuk dikembangkan lebih lanjut. PHP dapat diintegrasikan *(embedde)* ke dalam web *server*, atau dapat berperan sebagai program CGI yang terpisah. Karakteristik yang paling unggul dan paling kuat dalam PHP adalah lapisan integrasi *database* (*database integration layer*). *Database* yang didukung PHP adalah: *Oracle, Adabas-D, Sybase, FilePro, mSQL, Velocis, MySQL, Informix, Solid, dBase, ODBC, Unix dbm, dan PostgreSQL*[6].

Tujuan utama penggunaan bahasa ini adalah untuk memungkinkan perancang web menulis halaman web dinamik dengan cepat[6].

2.2.6.1.1 Framework Codeigniter

Framework adalah kumpulan perintah atau fungsi dasar yang membentuk aturan-aturan tertentu dan saling berinteraksi satu sama lain sehingga dalam pembuatan aplikasi website kita harus mengikuti aturan dari framework tersebut[6].

Dengan framework (dalam hal ini framework php), kita tidak perlu memikirkan kode perintah/fungsi dasar dari aplikasi website kita. Seperti bagaimana mengambil data dari database untuk ditampilkan. Kita hanya memikirkan apa kode sql-nya dan ditampilkan kemana? Hal-hal penunjang lainnya seperti koneksi database validasi form, GUI, dan keamanan; telah disediakan oleh framework sehingga jumlah baris kode yang kita buat jauh lebih sedikit dibandingkan jika semua kode dari kita[6].

Jadi, keuntungan yang dapat diperoleh dari pengguna framework adalah:

1. Waktu pembuatan aplikasi website kita jauh lebih singkat.

2. Kode aplikasi website menjadi lebih mudah dibaca, karena sedikit dan sifatnya pokok.Detailnya adalah kode dari framework dan ini mungkin tidak perlu dipikirkan,terjamin.
3. Website kita menjadi lebih mudah diperbaiki,karena kita tidak perlu fokus ke semua komponen kode website, terutama kode system framework.
4. Kita tidak perlu lagi membuat kode penunjang aplikasi website seperti koneksi database,validasi form,GUI,dan keamanan.
5. Pikiran kita menjadi lebih terfokus ke kode alur permasalahan website, apa yang ditampilkan dan layanan apa saja yang diberikan dari aplikasi website tersebut.
6. Jika dikerjakan team work, maka akan lebih terarah karena system framework, mengharuskan adanya keteraturan peletakan kode. seperti bagian pengambilan database terpisah dengan bagian pengaturan tampilan untuk pengunjung[6].

Jadi intinya banyak keuntungan yang dapat diperoleh, terutama bagi *developer web* yang selalu dihantui waktu *deadline* proyek *website*-nya. Tidak selesai waktunya sesuai target, klien akan kecewa dan mungkin akan lari ke developer lain. Jika anda bersikeras membuat *website* dari nol kode maka akan butuh banyak waktu dan pasti akan ketinggalan dengan teman anda yang menggunakan *framework*[6].

Framework CI paling mudah dikuasai untuk seorang pemula, karena help-nya cukup tersedia diwebsitenya. bahkan saat kita men-*download framework* ini dari internet, sudah disertai *help*-nya yang mudah dimengerti, mulai dari langkah instalasi sampai fungsi-fungsinya tersedia. Adanya *help*/dokumentasi sangat mudah bagi pemula dalam mempelajari lingkungan pengembangan *website* dengan CI. *Framework* Codeigniter dapat dioperasikan dalam PHP 4.3.2 + maupun 5 sehingga kita membuat aplikasi *website* pada sebuah *server* yang masih belum *support* PHP5, ini tidak akan menjadi masalah,karena masih tetap dapat berkerja[6].

MVC (*Model View Controller*) adalah sebuah pola program yang bertujuan memisahkan logika bisnis,logika data, dan logika tampilan (*interface*), atau secara sederhana memisahkan antara proses, data dan tampilan mvc mengatur arsitektur sebuah aplikasi. umumnya apliksai yang dibangun dengan konsep MVC adalah aplikasi yang dibangun dengan konsep MVC adalah aplikasi yang cukup besar, karena salah satu keuntungan dari MVC itu adalah kemudahan *maintenance*, dan pengembangan aplikasi tersebut. CI menggunakan konsep ini anda harus memisahkan kode database ke folder model,kode proses ke folder *controller* dan kode tampilan ke *folder view*[6].

2.2.6.2 Android

Android merupakan suatu *software stack* untuk *mobile device*. Di dalamnya terdapat sistem operasi, *middleware*, dan *key application*. Aplikasi pada *platform* ini dikembangkan dalam bahasa pemrograman java[12].

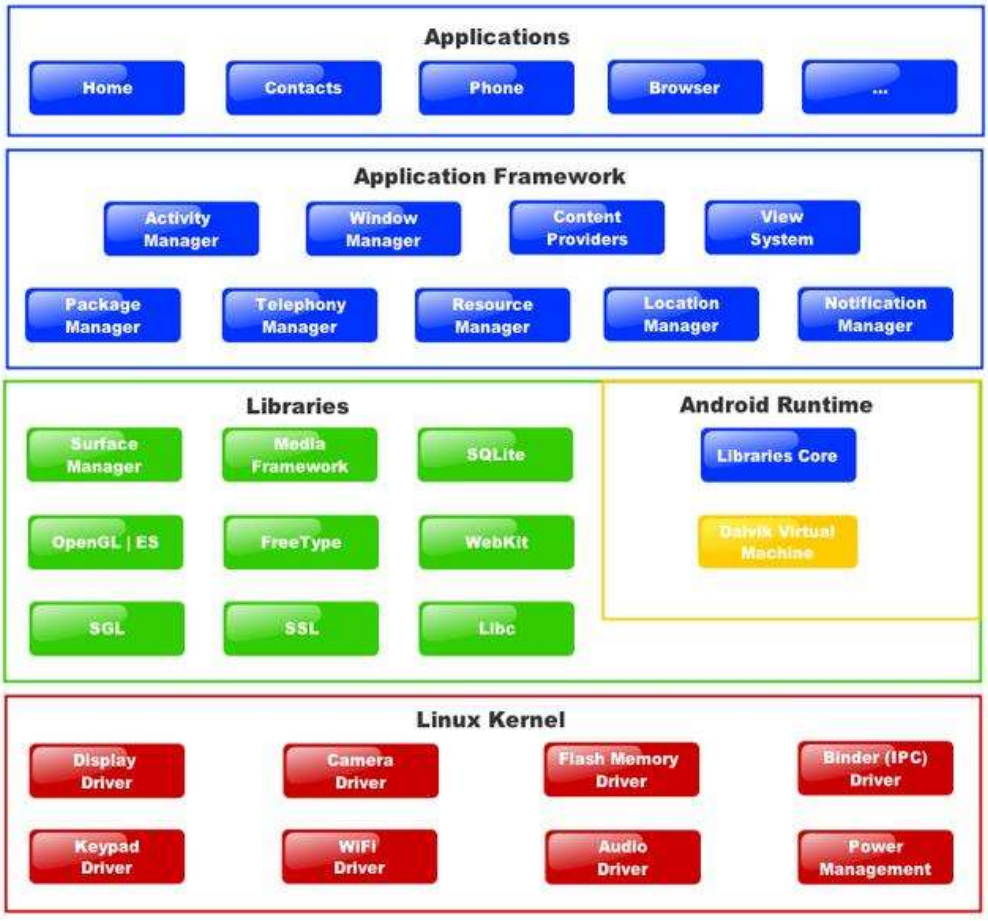
Android sendiri memiliki banyak fitur diantaranya adalah:

1. Merupakan sebuah *Application Framework* sehingga *programmer* dapat menggunakan beberapa fungsi yang telah disediakan.
2. *Dalvik virtual machine*. Tiap aplikasi dalam Android memiliki *instance virtual machine* yang dapat bekerja secara efisien dalam lingkungan memoriyang terbatas.
3. *Integrated browser*. *Web browser* berbasis WebKit *engine* terdapat pada *browser default* Android atau pun dapat diintegrasikan dengan aplikasi lain.
4. *Optimized graphics*. *Library grafis* 2D yang kaya dan 3D berbasis OpenGL ES 1.0 yang mendukung akselerasi *hardware*.
5. SQLite. Basis data relasional yang ringan namun sangat *powerful*.
6. *Media Support*. Mendukung berbagai format audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
7. GSM Telephony. Mendukung fungsi komunikasi GSM.
8. Bluetooth,EDGE,3G dan WiFi. Mendukung komunikasi pada jaringan (tergantung *hardware*).

9. Kamera, GPS, kompas dan *accelerometer*. Mendukung berbagai fitur yang disediakan oleh *hardware*.
10. Kakas pengembangan yang lengkap. Termasuk *device emulator*, *tools* untuk *debugging*, *profiling* memori dan performa, *plugin* untuk Eclipse IDE[12].

2.2.6.2.1 Arsitektur Android

Android terdiri dari beberapa stack software yang terdiri dari: Applications, *Application Framework*, *Libraries*, *Android Runtime* dan *Kernel Linux*. Arsitektur lengkap platform ini dapat dilihat pada Gambar 2.3.



Gambar 2.4 Arsitektur platform android

2.2.6.2.2 *Application*

Application merupakan program yang langsung berhubungan dengan user. Baik program yang merupakan bawaan dari Android sendiri maupun program

yang dibuat oleh developer menggunakan bahasa pemrograman java. Contoh program bawaan dari *platform* Android sendiri adalah *email client*, program SMS, *calendar*, maps, *web browser*, *contact* dan sebagainya[12].

2.2.6.2.3 *Application Framework*

Lapisan ini berisi sekumpulan API yang dapat digunakan oleh *programmer* maupun *core application* dari Android. Lapisan ini dirancang untuk memudahkan penggunaan komponen dari Android sendiri. Aplikasi manapun dalam Android dapat berbagi fungsi sehingga aplikasi lain dapat memanfaatkannya[12].

Aplikasi pada Android disusun atas beberapa komponen:

1. Sekumpulan *Views*. Digunakan untuk mengatur tampilan pada aplikasi. Contohnya adalah *lists*, *grids*, *text box*, *button*, bahkan *embeddable web browser*.
2. *Content providers*. Komponen yang mengatur agar aplikasi dapat mengakses *resources* dari aplikasi lain (seperti *Contacts*), atau berbagi data dengan aplikasi lain.
3. *Resource Manager*. Menyediakan akses ke pada *resource non-code* seperti *localized string*, *grafik* dan *file layout*.
4. *Notification Manager*. Memungkinkan agar suatu aplikasi dapat menampilkan peringatan yang dapat di kostumasi pada status bar.
5. *Activity Manager*. Mengatur siklus aplikasi dan navigasi antar aplikasi yang sedang berjalan.

2.2.6.2.4 *Libraries*

Android mendukung beberapa *library* C/C++ yang digunakan pada berbagai komponen Android. Kemampuan ini dapat diakses oleh *developer* melalui Android *application framework*. Beberapa *library* diantaranya adalah[[12]]:

1. *System* C *library*. Implementasi *library* C standar (libc).
2. *Media Libraries*. Mendukung berbagai format multimedia (termasuk
3. MPEG4,H.264, MP3, AAC, AMR, JPG, PNG).
4. *Surface Manager*. Mengatur akses ke *subsistem display*.
5. LibWebCore. *Engine web browser modern*.
6. SGL. *Engine* grafis 2D.

7. 3D *Library*. Implementasi OpenGL ES 1.0 yang mendukung akselerasi *hardware*.
8. *FreeType*. *Rendering* untuk bitmap dan vector *font*.
9. SQLite. Basis data relasional yang kecil namun sangat ampuh.

2.2.6.2.5 *Android Runtime*

Tiap aplikasi pada Android memiliki proses-nya masing-masing. Tiap aplikasi tersebut memiliki instans dari *Dalvik virtual machine* (VM). Dalvik virtual machine dirancang agar suatu *device* dapat menjalankan beberapa VM secara efisien. Dalvik VM mengeksekusi file dengan format Dalvik *Executable* format (.dex) yang dirancang untuk meminimalkan *memory footprint* . Dalvik VM berbasis register, dan dapat menjalankan kelas-kelas yang dikompilasi dengan bahasa pemrograman java dan ditransformasikan menjadi format .dex. Dalvik VM sendiri bergantung pada Kernel Linux untuk fungsi dasarnya, seperti threading manajemen memori secara low-level[12].

2.2.6.2.6 Linux Kernel

Android menggunakan Kernel Linux versi 2.6 sebagai sistem utama. Fungsi kernel yang digunakan antara lain untuk keamanan, manajemen memori, manajemen proses, manajemen jaringan dan driver model. Kernel juga berfungsi sebagai layer abstrak antara hardware dan lapisan lainnya pada software stack[12].

2.2.6.2.7 Anatomi Aplikasi Android

Terdapat empat building blocks pada aplikasi Android yaitu:

1. *Activity*
2. *Intent Receiver*
3. *Service*
4. *Content Provider*

Tidak semua aplikasi membutuhkan keempat blok ini, tetapi suatu aplikasi dibuat menggunakan kombinasi beberapa blok ini. Setelah memutuskan blok komponen yang akan digunakan, blok tersebut didaftarkan pada suatu file yang disebut dengan AndroidManifest.xml. File XML ini digunakan untuk menyatakan komponen apa saja yang dibutuhkan oleh sebuah aplikasi dan kemampuan serta kebutuhan aplikasi tersebut[12].

2.2.6.2.8 *Activity*

Activity paling umum diantara keempat *building block*. *Activity* biasanya suatu tampilan (screen) dalam suatu aplikasi. Tiap activity diimplementasikan sebagai satu kelas yang meng-*extend base class Activity*. Kelas yang dibuat akan menampilkan *user interface* yang terdiri dari beberapa *views* dan akan merespon suatu *event*. Kebanyakan, aplikasi terdiri dari beberapa *screen*. Contohnya aplikasi pengiriman pesan akan memiliki screen untuk mendaftarkan pesan yang masuk, menulis pesan dan *screen* lainnya untuk melakukan pengaturan. Tiap *screen* pada contoh ini akan diimplementasi sebagai suatu *activity*. Aplikasi dapat berpindah ke screen lain dengan cara memulai suatu activity baru. Pada suatu kasus tertentu, activity akan mengembalikan nilai kepada activity sebelumnya. Contohnya activity untuk memilih foto akan mengembalikan foto yang dipilih pada *activity* yang memanggilnya. Pada saat *screen* baru terbuka, screen sebelumnya akan dihentikan dan *state* nya akan disimpan pada *history stack*. *User* dapat melakukan navigasi ke *activity* sebelumnya melalui *history* tersebut. Screen akan di hapus dari *history stack* apabila sistem memutuskan bahwa screen tersebut tidak layak untuk berada pada memori. Android menyimpan *history stack* untuk setiap aplikasi yang dijalankan dari tampilan awal (*home screen*) [12].

2.2.6.2.9 Intent dan Intent Filter

Android menggunakan kelas khusus bernama intent untuk berpindah dari screen ke screen lainnya. Sebuah intent mendeskripsikan apa yang ingin aplikasi penuhi. Bagian struktur data dari intent yang paling penting adalah action dan data dari aksi yang dilakukan. Nilai action yang biasanya adalah MAIN, VIEW, PICK, EDIT dan sebagainya. Data diekspresikan sebagai URI. Sebagai contoh, untuk melihat informasi suatu contact seseorang, kita membuat intent dengan action VIEW dan data sebagai URI yang merepresentasikan orang tersebut. Terdapat suatu kelas yang dinamakan IntentFilter. Apabila intent adalah suatu permintaan untuk melakukan sesuatu, IntentFilter adalah deskripsi yang dapat dilakukan oleh intent activity. Activity yang dapat menampilkan informasi contact dari seseorang akan mempublikasikan IntentFilter yang menyatakan bahwa ia dapat menangani action VIEW apabila dipakai pada data yang merepresentasikan seseorang. Suatu

aplikasi mempublikasikan IntentFilter pada file AndroidManifest.xml. Navigasi dari suatu screen ke screen lainnya dipenuhi dengan cara me-resolve intent. Untuk melakukan navigasi ke depan, activity memanggil startActivity(myIntent). Sistem akan mencari IntentFilter untuk semua aplikasi dan memilih activity yang IntentFilter-nya yang paling mendekati dengan myIntent. activity baru akan mendapat informasi tentang intent yang menyebabkan activity tersebut dijalankan. Proses pencarian intent terjadi sewaktu runtime pada saat startActivity dipanggil. Hal tersebut membawa keuntungan sebagai berikut[12]:

1. *Activity* dapat melakukan *reuse functionality* dari komponen lain dengan cara melakukan permintaan dalam bentuk intent.
2. *Activity* dapat di *replace* kapan pun dengan activity yang ekuivalen.

2.2.6.2.10 Intent Recivier

Intent Receiver dapat digunakan pada saat kita menginginkan kode kita dijalankan akibat reaksi dari *external event* tertentu. Contohnya, pada saat telepon berbunyi, atau pada saat ada data pada jaringan, atau pada suatu waktu tertentu. *Intent Receiver* tidak menampilkan *user interface*, meskipun ia dapat menggunakan *Notification Manager* untuk memperingatkan *user* apabila sesuatu terjadi. *Intent Receiver* didaftarkan pada file AndroidManifest.xml, tetapi secara manual juga bisa ditambahkan dengan fungsi Context.registerReceiver(). Suatu aplikasi tidak harus berjalan pada IntentReceivernya agar bisa dipanggil, tetapi sistem yang akan menjalankan aplikasi tersebut. Apabila diperlukan, pada saat Intent *Receiver* mendapat *trigger*, aplikasi juga dapat mengirimkan intent broadcast kepada aplikasi lainnya dengan fungsi Context.broadcastIntent()[12].

2.2.6.2.11 *Service*

Service adalah sebuah kode yang hidup terus-menerus dan berjalan tanpa memiliki tampilan. Contoh yang sederhana adalah suatu aplikasi media player yang sedang memainkan lagu dari *play list*. Pada aplikasi media player, kemungkinan terdapat satu atau lebih *activity* yang memperbolehkan user memilih lagu dan memainkannya. Tetapi musik yang sedang diputar tidak boleh ditangani oleh activity karena user akan menginginkan musik tetap diputar meskipun ia berpindah ke screen baru. Pada kasus ini *activity* media player dapat menjalankan

service menggunakan fungsi Context.startService() untuk menjalankan *service* di background yang memutar musik. Sistem akan tetap memutar musik sampai selesai. Selain itu *activity* dapat terhubung dengan service menggunakan fungsi Context.bindService(). Pada saat terhubung, aplikasi dapat berkomunikasi melalui *interface* yang diberikan oleh *service* yang bersangkutan. Pada contoh media *player*, *interface service* akan memberikan fungsi *pause*, *rewind* dan sebagainya[12].

2.2.6.2.12 *Content Provider*

Content provider digunakan untuk pertukaran data antara aplikasi. Data yang dipertukarkan bersifat *shared*. *Content provider* adalah sebuah kelas yang berisi method standar yang memungkinkan aplikasi lain dapat menyimpan dan mengambil suatu data yang ditangani oleh *content provider*[12].

2.2.6.2.13 Siklus Hidup Aplikasi Android

Pada banyak kasus, tiap aplikasi pada Android masing-masing memiliki Linux proses. Proses ini diciptakan untuk aplikasi tersebut pada saat kode program tersebut akan dieksekusi, akan terus berjalan sampai tidak dibutuhkan lagi dan sistem memerlukan memori untuk aplikasi lain. Konsep dasar program Android adalah suatu proses pada aplikasi tidak langsung di atur oleh aplikasi tersebut. Proses tersebut ditentukan oleh sistem melalui kombinasi : informasi aplikasi bagi sistem, seberapa pentingnya aplikasi tersebut bagi user dan ketersediaan memori[12].

Contoh kasus life-cycle bug adalah IntentReceiver yang menciptakan *thread* pada saat menerima *intent* pada *method* onReceiveIntent() dan selesai dari fungsi tersebut. Setelah fungsi tersebut selesai, sistem menganggap bahwa *Intent Receiver* tersebut menjadi tidak aktif sehingga ia memiliki proses yang tidak dibutuhkan (kecuali terdapat komponen aplikasi lain di dalamnya). Oleh karena itu, sistem dapat menghapus proses tersebut kapanpun apabila dibutuhkan. Hal tersebut akan menghapus semua *thread* yang berjalan pada proses tersebut. Solusi masalah ini adalah memakai *Service* dari *Intent Receiver*, sehingga sistem mengetahui bahwa ada suatu yang aktif dalam proses tersebut. Sebagai

programmer, kita harus mengetahui bagaimana cara yang tepat untuk merancang aplikasi.

Kesalahan pada perancangan akan membuat suatu aplikasi akan dihentikan oleh sistem pada saat yang tidak diinginkan. Untuk menentukan proses mana yang akan dihapus pada situasi low memory, Android memiliki "importancehierarchy" berdasarkan komponen dan state yang sedang berjalan. Urutan berdasarkan kepentingan tersebut adalah[12]:

1. *Foreground process*. Adalah suatu proses yang dibutuhkan untuk berinteraksi dengan user saat ini. Berbagai komponen suatu aplikasi dapat mengakibatkan proses di dalamnya menjadi foreground proses. Suatu proses dikatakan foreground apabila memenuhi kondisi sebagai berikut:
   1.1 Menjalankan *Activity* pada *top screen* yang sedang berinteraksi dengan user (setelah method onResume() dipanggil).
   1.2 Mempunyai IntentReceiver yang sedang berjalan (*method*
   1.3 IntentReceiver.onReceiveIntent() sedang dieksekusi).
   1.4 Memilik *service* yang fungsi *callback*-nya (Service.onCreate(), Service.onStart() atau Service.onDestroy()) sedang dieksekusi.
   1.5 Hanya akan terdapat beberapa proses seperti itu dalam sistem, proses ini hanya akan dihapus apabila sistem tidak memiliki pilihan lain. Contohnya apabila memori tinggal sedikit sehingga proses *foreground* tidak bisa lagi berjalan. Hal ini diperlukan untuk menjaga *user interface* tetap interaktif dengan user.
2. *Visible process*. Adalah proses yang memegang *Activity* yang dapat dilihat *user* pada layar tetapi tidak *foreground* (pada saat *method* onPause() dipanggil). Hal ini dapat terjadi, sebagai contoh *activity foreground* muncul dengan dialog sehingga activity sebelumnya terlihat di belakangnya. Proses ini sangat penting dan tidak akan di hapus kecuali untuk menjaga aplikasi *foreground* tetap berjalan.
3. *Service process*. Adalah proses yang memegang *Service* yang dimulai dengan method StartService().Meskipun proses ini secara langsung tidak dapat dilihat oleh *user*, biasanya proses ini menjalankan perintah dari user (seperti

memutar lagu, men-download data dari jaringan dan lain-lain). Sistem akan tetap membiarkan proses ini berjalan selama proses *foreground* dan *visible* masih bisa berjalan.

4. *Background process*. Adalah proses yang memegang Activity yang saat ini tidak dapat dilihat oleh user (*method* onStop() sedang dipanggil). Proses ini tidak secara langsung mempengaruhi user. Apabila *activity life cycle* telah dipenuhi dengan benar, sistem dapat menghapus proses ini sewaktu-waktu untuk memberikan memori tambahan. Biasanya banyak ditemukan *proses* semacam ini, sehingga proses tersebut disimpan dalam LRU (*Least RecentlyUsed*) untuk menjaga agar proses yang baru saja dilihat oleh user akan dihapus paling akhir pada kondisi *low memory*.
5. *Empty Process*. Adalah proses yang tidak memegang suatu komponen yang sedang aktif. Alasan mengapa membiarkan proses semacam ini adalah untuk melakukan cache sehingga proses startup menjadi lebih cepat apabila aplikasi tersebut akan dijalankan kembali. Sistem seringkali menghapus proses semacam ini untuk mencapai keseimbangan dari system resource antara *cached processes* dan kernel *caches*.

2.2.6.2.14 *Tools* Implementasi

Tools implementasi merupakan peralatan-peralatan yang dibutuhkan dalam membangun perangkat lunak. Adapun tools implementasi yang digunakan untuk membuat aplikasi pemesanan makanan pada restoran adalah JDK sebagai perangkat pembangun Java, Eclipse IDE sebagai lingkungan pengembangan perangkat lunak, dan SQL sebagai sistem manajemen basis data[12].

2.2.6.2.15 Java Development Kit

Java Development Kit (JDK) adalah sebuah perangkat peralatan yang digunakan untuk membangun perangkat lunak dengan menggunakan bahasa pemrograman Java. JDK berjalan diatas sebuah virtual machine yang dinamakan JVM (Java Virtual Machine). Dokumentasi JDK berisi spesifikasi API, deskripsi fitur, panduan pengembang, referensi halaman untuk perkakas JDK dan utilitas, demo, dan link ke informasi terkait. Dokumentasi ini juga tersedia dalam bundel yand dapat diunduh dan diinstal pada *computer*[12].

2.2.6.3 Eclipse IDE

Eclipse adalah sebuat komunitas proyek open source yang difokuskan pada membangun sebuah platform pengembangan yang dapat diperluas, runtime dan kerangka aplikasi untuk membangun, menyebarkan dan mengelola perangkat lunak di seluruh daur hidup perangkat lunak. Komunitas Eclipse memiliki lebih dari 60 proyek open source. Proyekproyek ini secara konsep terbagi menjadi 7 kategori [12]:

1. *Enterprise Development*
2. *Embedded and Device Development*
3. *Rich Client Platform*
4. *Rich Internet Applications*
5. *Application Frameworks*
6. *Application Lifecycle Management (ALM)*
7. *Service Oriented Architecture (SOA)*

Komunitas Eclipse juga didukung oleh ekosistem besar dan dinamis dari penyedia solusi teknologi informasi utama, start-up inovatif, universitas dan lembaga penelitian, dan individu yang memperluas, mendukung, dan melengkapi platform Eclipse[12].

2.2.6.4 *XAMPP*

XAMPP Kepanjangan dari *XAMPP* yaiu *APACHE*, PHP, *MySQL* dan PHPMyAdmin. *XAMPP* merupakan tool yang menyediakan paket perangkat lunak ke dalam satu buah paket. Dengan menginstall *XAMPP* maka tidak perlu lagi melakukan instalasi dan konfigurasi web *server APACHE*, PHP dan *MySQL* secara manual. *XAMPP* akan menginstalasi dan mengkonfigurasikannya secara otomatis untuk anda atau auto konfigurasi.

2.2.6.5 *APACHE*

APACHE merupakan salah satu jenis program yang bertujuan untuk mengirimkan kembali sintaks yang telah diberikan oleh user dan menampilkan kembali dalam bentuk dalam format hasilnya saja. Program *APACHE* ini hanya terdapat pada sebuah *server* yang terdapat pada ISP (Internet Service Provider) atau di Indonesia disebut Jasa Penyelenggara Internet. Saat ini program yang

paling banyak terdapat pada *server-server* di dunia adalah *APACHE*. Hal ini disebabkan karena beberapa keunggulan yang diberikan.

Keunggulan-keunggulan tersebut antara lain :

1. Memiliki kemapuan pembacaan yang tinggi dan mudah disesuaikan
2. Memiliki tingkat keamanan yang tinggi. Hal ini dibutuhkan bagi para penyimpan data yang memiliki tingkat kerahasian tertentu.
3. Dapat dijalankan untuk berbagai macam operasi sistem.

APACHE merupakan suatu nama yang diambil dari istilah "*A PAtCHy*" *server* yang dapat diartikan dengan mengambil suatu patch atau bagian dari program yang diberikan.

APACHE sudah berkembang sejak versi pertamanya. Sampai saat ditulisnya artikel ini versi terakhirnya yang ada yaitu *APACHE* ver 2.0.54. *APACHE* bersifat open source, artinya setiap orang boleh menggunakannya, mengambil dan bahkan mengubah kode programnya.

Tugas utama *APACHE* adalah menghasilkan halaman web yang benar kepada peminta, berdasarkan kode PHP yang dituliskan oleh pembuat halaman web. Jika diperlukan juga berdasarkan kode PHP yang dituliskan, maka dapat saja suatu *database* diakses terlebih dahulu (misalnya dalam *MySQL*) untuk mendukung halaman web yang dihasilkan.

2.2.6.6 *MySQL*

Perkembangannya disebut *SQL* yang merupakan kepanjangan dari *Structured Query Language*. *SQL* merupakan bahasa terstruktur yang khusus digunakan untuk mengolah *database*. *SQL* pertama kali didefinisikan oleh *American National Standards Institute* (ANSI) pada tahun 1986. *MySQL* adalah sebuah sistem manajemen *database* yang bersifat open source. *MySQL* adalah pasangan serasi dari PHP. *MySQL* dibuat dan dikembangkan oleh *MySQL* AB yang berada di Swedia[10].

MySQL dapat digunakan untuk mengelola *database* mulai dari yang kecil sampai dengan yang sangat besar. *MySQL* juga dapat menjalankan perintah-perintah *Structured Query Language* (*SQL*) untuk mengelola *database-database* yang ada di dalamnya. Hingga kini, *MySQL* sudah berkembang hingga versi 5.

MySQL 5 sudah mendukung *trigger* untuk memudahkan pengelolaan tabel dalam *database*[10].

Sebuah *Website* yang interaktif dan dinamis, tentunya akan membutuhkan penyimpanan data yang fleksibel dan cepat untuk diakses. Salah satu *database* untuk *server* adalah *MySQL*. *MySQL* adalah merupakan aplikasi RDBMS (*Relational Data Base Management System*) yang sangat cepat dan dapat diandalkan dalam menangani basis data. Jenis *database* ini sangat popular dan digunakan pada banyak website . *MySQL* menggunakan bahasa *SQL* (*Struktur Query Language*) dan bersifat *free* (gratis atau tidak perlu membayar untuk menggunakannya). Selain itu, *MySQL* dapat berjalan di berbagai platform, antara lain Linux, Windows, dan lain sebagainya[10].

Melakukan koneksi kepada *database MYSQL* dengan menggunakan PHP menggunakan dua tahap :

1. Koneksi kepada *server MySQL*

   Dilakukan untuk melakukan otentikasi, digunakan untuk login ke dalam *server*, menggunakan *MySQL*_connect( ).
2. Koneksi kepada *database*

   Dilakukan untuk memilih *database* mana yang ada dalam *server* yang akan dimanipulasi, karena bisa saja *database* yang dimaksud sudah tidak ada. Fungsi yang digunakan adalah *MySQL*_select_db( ).

2.2.6.6.1 Pengertian *MySQL*

MySQL adalah perangkat lunak *database server* atau sebut saja *Database Smart*. *Database* ini semakin lama semakin populer. Dengan menggunakan *database* ini, data semakin aman dan berdaya guna. *Database* ini juga banyak dipakai pada web *database* sehingga data semakin terintegrasi antara *database* dekstop dengan *database* web. Untuk menggunakan *database MySQL* harus menginstalasinya dahulu ke komputer[10].

2.2.6.6.2 Keistimewaan *MySQL*

Sebagai *server database* dengan konsep *database* modern, *MySQL* memiliki keistimewaan. Beberapa keistimewaan dimiliki *MySQL* sebagai berikut [10]:

1. *Portability*

   Database MySQL berfungsi dengan stabil tanpa kendala, berarti berlaku pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac OS X *Server*, Solaris, Amiga, HP-Unix, dan lain-lain

2. *Open Source*

   MySQL merupakan *database* open source (gratis), di bawah lisensi GPL sehingga dapat memperoleh dan menggunakannya secara cuma-cuma tanpa membayar sepersen pun.

3. *Multiuser*

   MySQL merupakan *database* yang dapat digunakan untuk menangani beberapa user dalam waktu bersamaan tanpa mengalami masalah. Dan memungkinkan sebuah *database server MySQL* dapat diakses client secara bersamaan pula.

4. *Performance Tuning*

   MySQL mempunyai kecepatan yang cukup baik dalam menangani query-query sederhana, serta mampu memproses lebih banyak *SQL* per satuan waktu.

5. *Column Type*

   Database MySQL didukung dengan tipe data yang sangat kompleks, seperti signed/unsigned *integer*, float, double, char, varchar, text, blob, data, time, datetime, timestamp, year, set serta enum

6. *Command And Functions*

   MySQL server memiliki operator dan fungsi secara penuh yang mendukung perintah SELECT dan WHERE dalam query.

7. *Security*

   Sistem *Security* pada *MySQL* mempunyai beberapa lapisan sekuritas seperti tingkatan *subnetmask*, *hostname*, dan izin akses user dengan sistem perizinan yang mendetil serta *password* terenkripsi.

8. *Scalability dan Limits*

   MySQL mempunyai kemampuan menangani *database* dalam skala cukup besar, dengan jumlah record lebih dari 50 juta dan 60 ribu tabel serta 5 miliar

baris. Selain itu dapat menampung indeks sampai 32 indeks pada tiap tabelnya.

9. *Connectivity*

   Adanya kemampuan *MySQL* melakukan koneksi dengan *client* menggunakan protokol TCP/IP, Unix socket (Unix), atau Named Pipes (NT).

10. *Localization*

    Adanya kemampuan dalam mendeteksi kesalahan (*error code*) pada *client* menggunakan lebih dari dua puluh bahasa.

11. *Interface*

    MySQL memiliki *interface* terhadap berbagai aplikasi dan bahasa pemograman menggunakan fungsi API (*Application Programming Interface*).

12. *Clients dan Tools*

    Database MySQL dilengkapi berbagai tools yang dapat digunakan untuk administrasi *database*.

13. Struktur Tabel

    MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE dibandingkan *database* lainnya.

2.2.6.7 PHP*MyAdmin*

Pengelolaan *database* dengan *MySQL* harus dilakukan dengan mengetikkan baris-baris perintah yang sesuai (*command line*) untuk setiap maksud tertentu. Jika ingin membuat *database*, pengguna harus mengetikkan baris perintah yang sesuai untuk membuat *database*. Tetapi hal tersebut tentu cukup menyulitkan karena pengguna harus hafal dan mengetikkan perintahnya satu persatu.

Banyak sekali perangkat lunak yang dapat dimanfaatkan untuk mengelola *database* dalam *MySQL*, salah satunya adalah PHP*MyAdmin*. Dengan PHP*MyAdmin* pengguna dapat membuat tabel, mengisi data dan lain-lain dengan mudah tanpa harus hafal perintahnya. Untuk mengaktifkan PHP*MyAdmin* langkah-langkahnya adalah : yang pertama setelah meng-*install XAMPP*, pengguna harus mengaktifkan web *server APACHE* dan *MySQL* dari control panel *XAMPP*. Yang kedua, jalankan di *browser* yang ada *(IE, Mozilla Firefox* atau *Opera)* lalu ketikkan alamat web berikut : http://localhost/PHPmyadmin/

pada address bar lalu tekan Enter. Langkah ketiga apabila telah nampak *interface* (tampilan antar muka) PHP*MyAdmin* pengguna bisa memulainya dengan mengetikkan nama *database*, nama tabel dan seterusnya.

2.2.6.8 Cascading Style Sheet (CSS)

1. Mengenal CSS

CSS merupakan bentuk kependekan dari *Cascading Style Sheets*, istilah ini apabila diartikan memiliki arti "hamparan pasir melata". Persepsi baru bahwa dokumen ini merupakan bentuk Script yang sangat membantu untuk membuat desain lebih menarik. CSS bukan merupakan bahasa pemograman, akan tetapi dapat disebut dengan spesial yang berguna untuk melakukan pengaturan pada setiap komponen (bagian-bagian seperti tabel, form, dan lain – lain) web. Inti dari dokumen CSS adalah Tag HTML, di dalam CSS tag HTML tidak lagi disebut sebagai Tag akan tetapi akan disebut sebagai Selector[6].

Fungsi terpenting dari dokumen CSS adalah melakukan pengaturan halaman web dengan sistem Style, karena dokumen CSS merupakan bentuk turunan dari elemen HTML (perintah html) maka dokumen ini merupakan bentuk kode yang embeded dengan Tag HTML, dalam penggunaannya kedua objek ini akan sangat berhubungan dan seakan tidak pernah terpisahkan[6].

2. Bagian Penting Dalam CSS

Dalam Penulisannya CSS memiliki sedikit perbedaan antara penulisan Tag HTML dengan Style CSS, telah dijelaskan bahwa pada penulisan CSS menggunakan Tag HTML, dan dalam dokumen CSS akan disebut dengan Selector. Secara garis besar, penulisan CSS sebagai berikut. :

```
Selector {
      Properti1: Value;
      Properti2: Value;
      }
```

Dari penggambaran penulisan CSS dapat dijelaskan bagian-bagiannya adalah[6]:

1. *Selector* : Bagian ini merupakan kumpulan dari Elemen HTML, jadi semua elemen yang ada pada HTML akan digunakan pada tag ini.
2. *Propertie*: Bagian ini merupakan artibut yang digunakan pada setiap elemen yang digunakan, tidak semua atribut akan dapat digunakan pada semua Selector, yaitu tergantung dukungan yang ada pada HTML.
3. *Value*: Merupakan nilai yang diberikan pada setiap Properti dalam Selector, penentuan nilainya dapat diset sama persis dengan pada saat menggunakan nilai pada atribut HTML.

2.2.6.9 Javascript (.js)

Javascript merupakan *cross-platform* yang dikembangkan oleh Netscape dan pertama kali digunakan dalam *browser* Netscape. Javascript dibuat agar mudah diintegrasikan kedalam program dan aplikasi lain, misalnya *browser*. Sebagian besar *browser* saat ini sudah mendukung javascript. Oleh karena itu, *script* dari javascript biasanya dimasukkan kedalam suatu HTML dan dieksekusi di*browser* lain[6].

Javascript bergantung kepada browser(navigator) yang memanggil halaman web yang berisi skrip skrip dari Javascript dan tentu saja terselip di dalam dokumen HTML. Javascript juga tidak memerlukan kompilator atau penterjemah khusus untuk menjalankannya (pada kenyataannya kompilator Javascript sendiri sudah termasuk di dalam browser tersebut)[6].